Figure 1

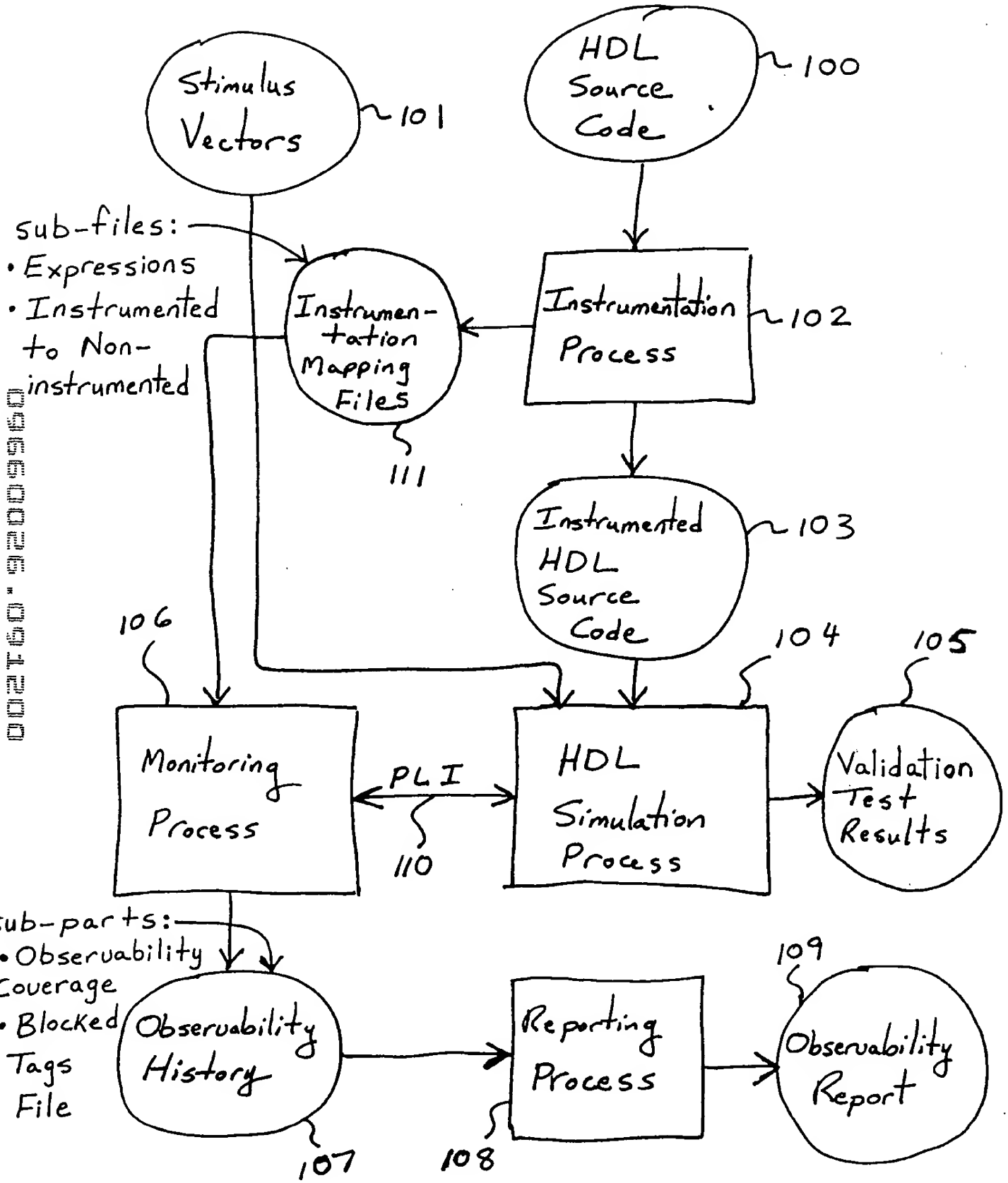
HDL Source Code
100
Stimulus Vectors
101
sub-files:
• Expressions
• Instrumented to Non-instrumented
Instrumentation Process
102
Instrumentation Mapping Files
111
Instrumented HDL Source Code
103
106
104
105
Monitoring Process
PLI
110
HDL Simulation Process
Validation Test Results
sub-parts:
• Observability Coverage
• Blocked Tags File
Observability History
107
Reporting Process
108
109
Observability Report

Figure 2

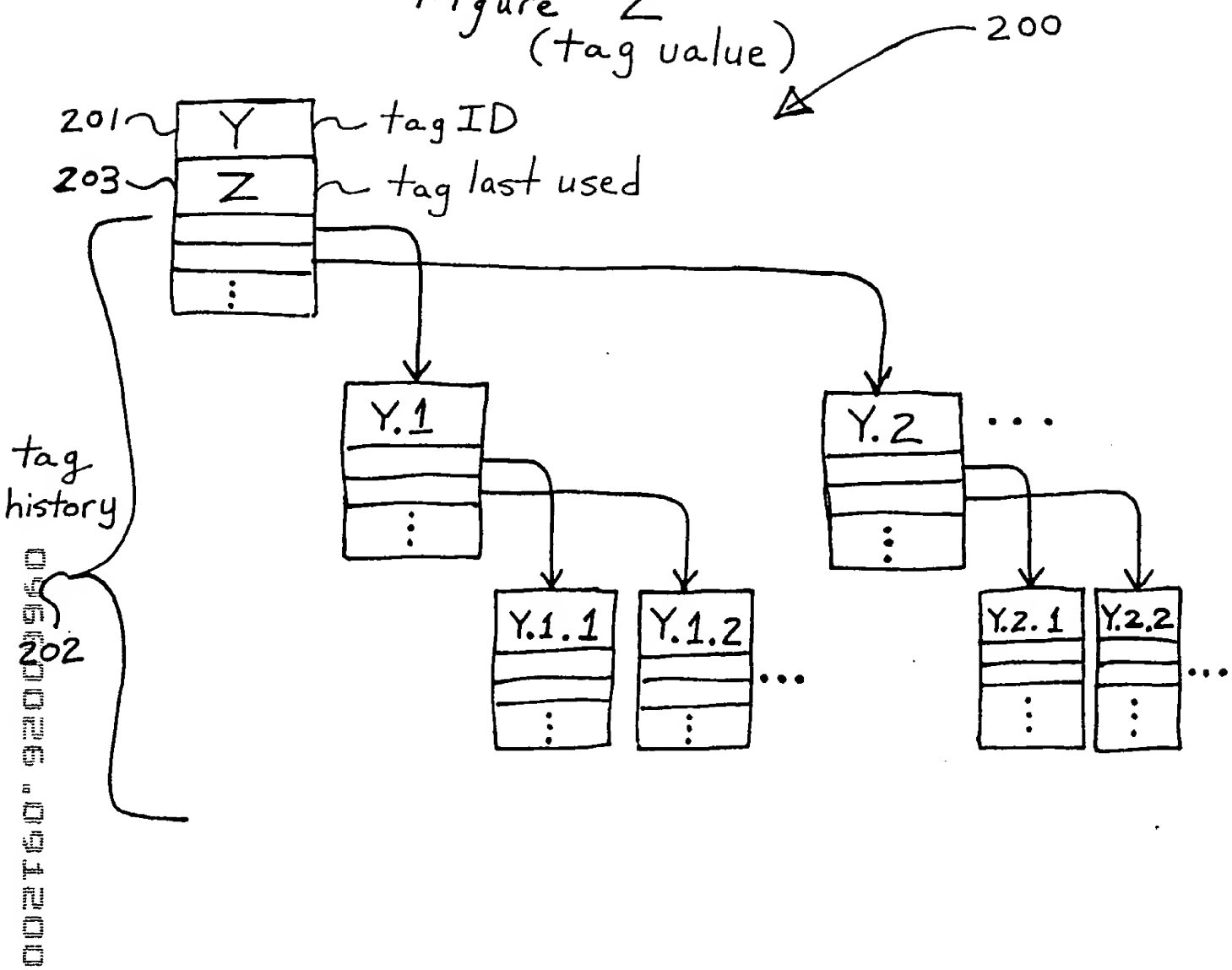
(tag value)
200
201
Y
tag ID
203
Z
tag last used
tag history
202
Y.1
Y.2
Y.1.1
Y.1.2
Y.2.1
Y.2.2

Figure 3A (abstracted assignment statement)

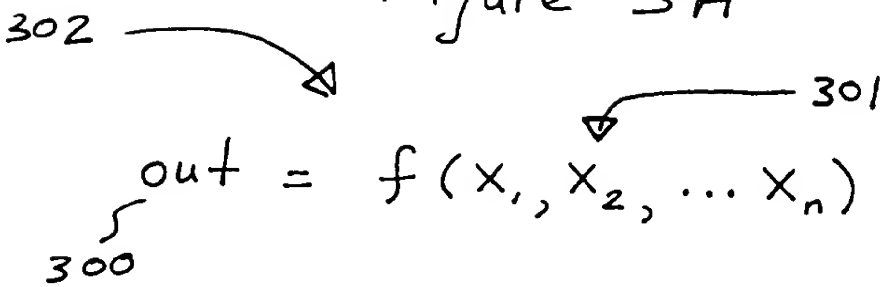

Figure 3B (tag value for abstracted assignment statement)

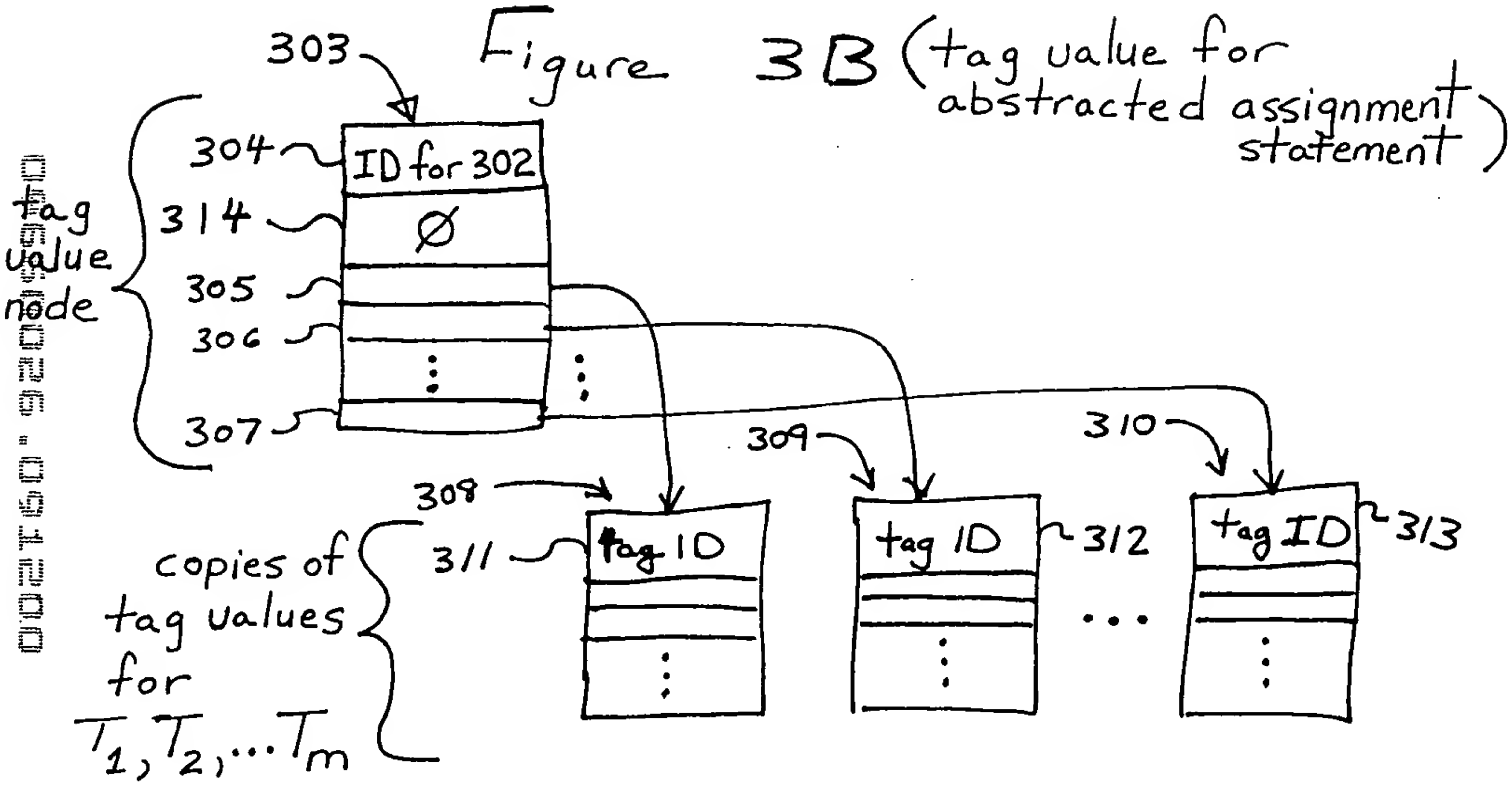

Figure 3C (example Verilog HDL assignment statement)

330

```
out = a & b;
```

Figure 3D

(example Verilog HDL assignment statement)

```
out_1  =  (in_3 & in_2) || (in_1 & in_0)
```

Figure 3E

(parse tree of rhs of assignment statement with rule-based tag propagation)

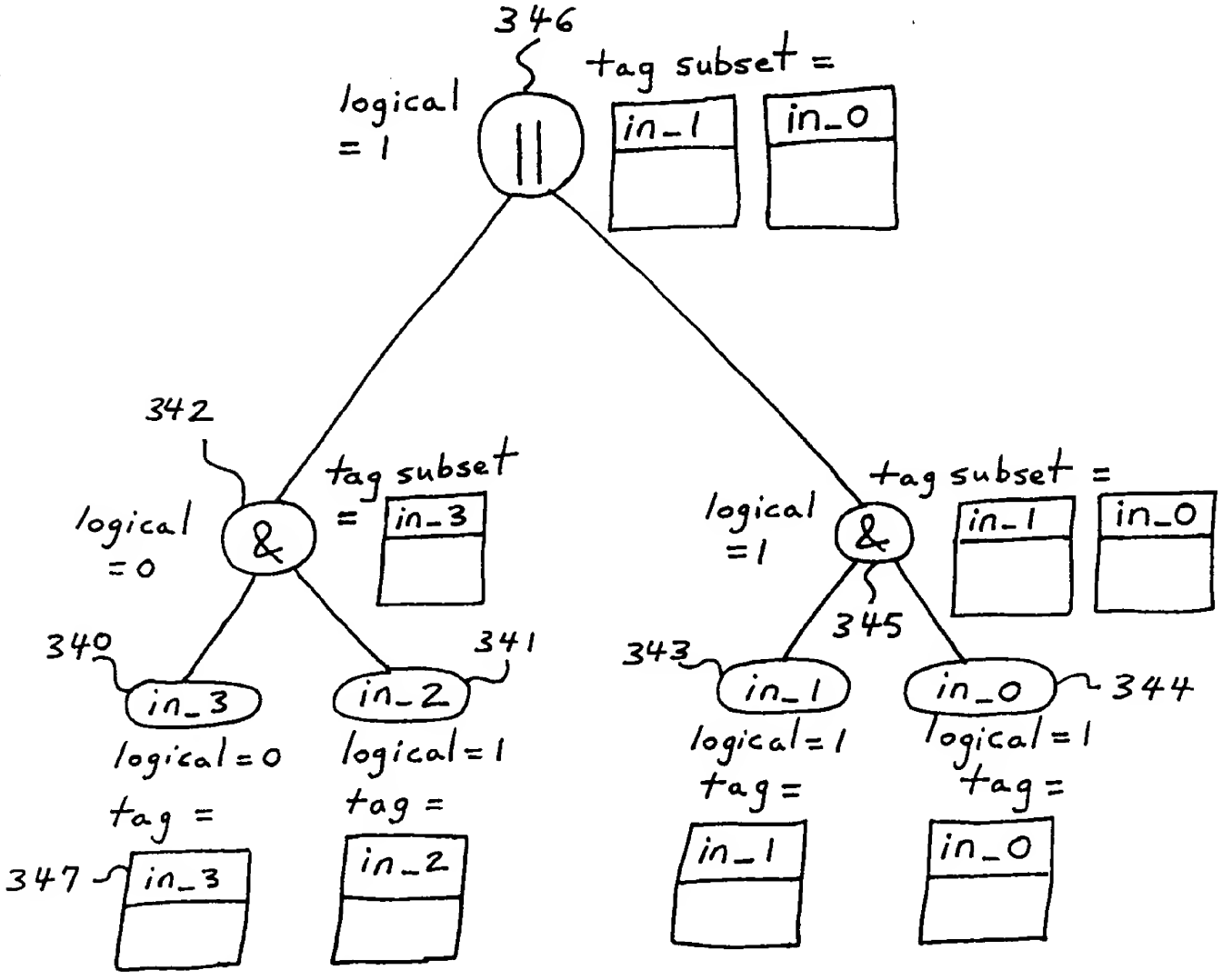

Figure 3F

(resulting tag value from rule-based propagation)

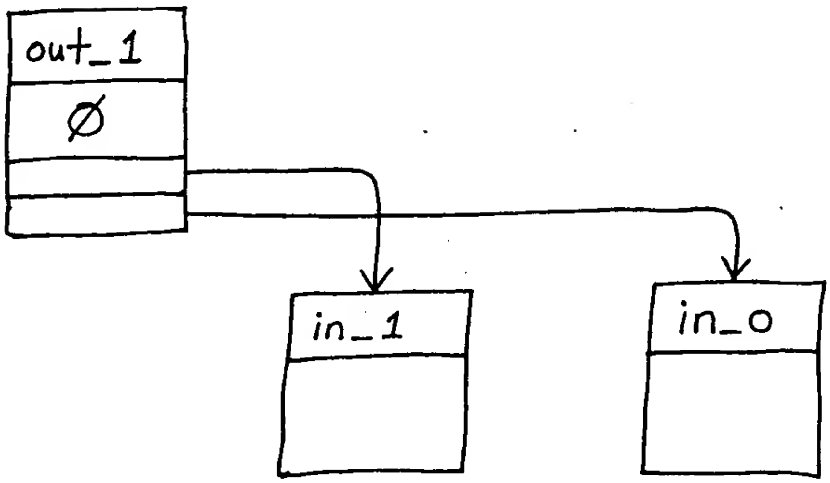

Figure 4 (abstracted conditionally executed assignment)

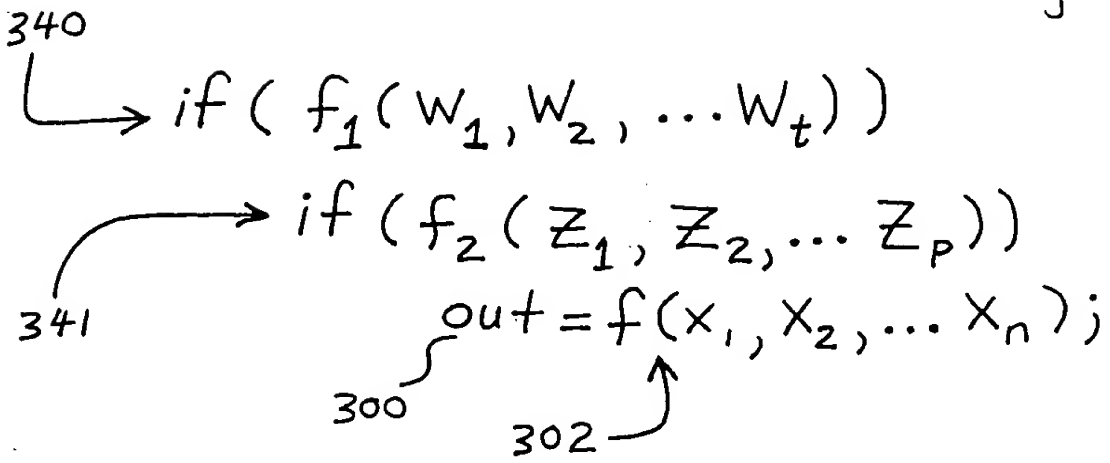

Figure 5A

(Verilog HDL program fragment with asynchronous assignments, an atomic block of synchronous assignments and conditionally executed assignments)

```
10      assign x = a & b;
20      assign y = a || c;

        aways @(posedge clk)
        begin
30              e = a & d;
40              f = b || h;
                if ( a & (c || d) )
                        if (x || y)
50                              e = f & g ;
                        else
60                              e = f || g;
        end
```

Figure 5B

(instrumented Verilog HDL program fragment)

```
10      assign x = a & b;
20      assign y = a || c;

        aways @(posedge clk)
        begin
30              $pli (30);
40              e = a & d;
50              f = b || h;
                if ( a & (c || d) )
                        if (x || y)
                                begin
60                                      $pli(60);
70                                      e = f & g ;
                                end
                        else
                                begin
80                                      $pli(80);
90                                      e = f || g;
                                end
        end
```

Figure 5C

Instrumented to Non-instrumented sub-file

10	10
20	20
30	40
40	50
50	70
60	90

Figure 5D

Expressions sub-file

PLI_call_instrumented_line_num 30
assign_stmt_instrumented_line_num 40
lhs_signal "e"
rhs_expression "a & d"

PLI_call_instrumented_line_num 30
assign_stmt_instrumented_line_num 50
lhs_signal "f"
rhs_expression "b || h"

PLI_call_instrumented_line_num 60
assign_stmt_instrumented_line_num 70
lhs_signal "e"
rhs_expression "f & g", "a&(c||d)", "x||y"

PLI_call_instrumented_line_num 80
assign_stmt_instrumented_line_num 90
lhs_signal "e"
rhs_expression "f || g", "a&(c||d)", "x||y"

PLI_call_instrumented_line_num 0
assign_stmt_instrumented_line_num 10
lhs_signal "x"
rhs_expression "a&b"

PLI_call_instrumented_line_num 0
assign_stmt_instrumented_line_num 20
lhs_signal "y"
rhs_expression "a || c"

Figure 5F

(hierarchical Verilog HDL program fragment)

```
/* Higher-level module */
module higher_mod;

        lower_mod lower_mod_inst(z, x, y & q)

endmodule


/* Lower-level modue */
module lower_mod (out1, in1, in2);

        assign out1 = in1 || in2;

endmodule
```

Figure 5G

```
Expressions sub-file
        PLI_call_instrumented_line_num 0
        assign_stmt_instrumented_line_num 0
        lhs_signal "lower_mod_inst.in2"
        rhs_expression "y & q"
```

Figure 5H

parameter passed to callback function when signal "y" changes
    PLI_call_instrumented_line_num 0
    assign_stmt_instrumented_line_num 0
    lhs_signal "lower_mod_inst.in2"
    rhs_expression "y & q"

parameter passed to callback function when signal "q" changes
    PLI_call_instrumented_line_num 0
    assign_stmt_instrumented_line_num 0
    lhs_signal "lower_mod_inst.in2"
    rhs_expression "y & q"

parameter passed to callback function when signal "x" changes
    PLI_call_instrumented_line_num 0
    assign_stmt_instrumented_line_num 0
    lhs_signal "lower_mod_inst.in1"
    rhs_expression "x"

parameter passed to callback function when signal "lower_mod_inst.out1" changes
    PLI_call_instrumented_line_num 0
    assign_stmt_instrumented_line_num 0
    lhs_signal "z"
    rhs_expression "lower_mod_inst.out1"

FIG. 6

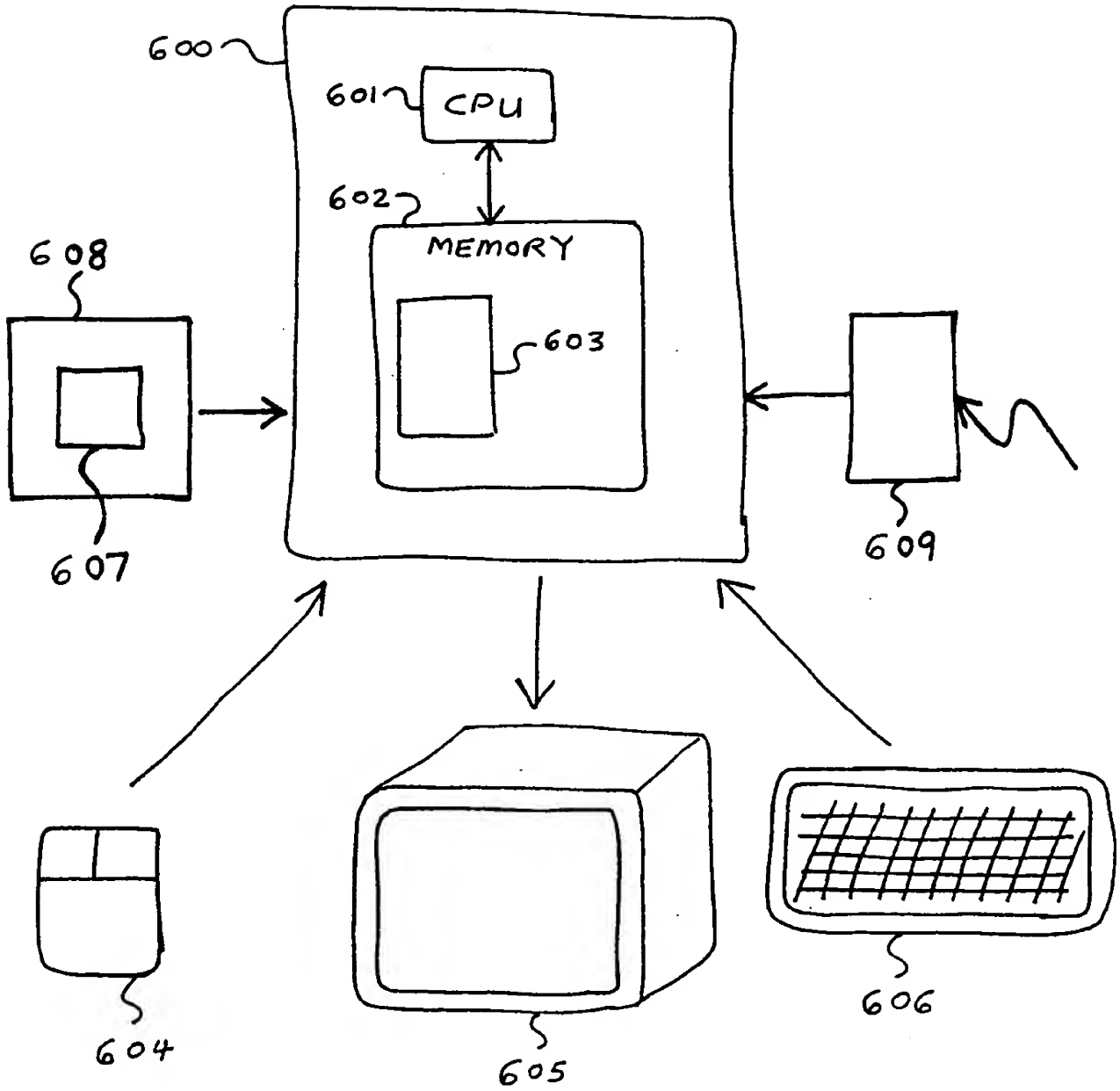

Figure 5E

parameter passed to callback function when signal "a" changes
PLI_call_instrumented_line_num 0
assign_stmt_instrumented_line_num 10
lhs_signal "x"
rhs_expression "a&b"

PLI_call_instrumented_line_num 0
assign_stmt_instrumented_line_num 20
lhs_signal "y"
rhs_expression "a || c"

parameter passed to callback function when signal "b" changes
PLI_call_instrumented_line_num 0
assign_stmt_instrumented_line_num 10
lhs_signal "x"
rhs_expression "a&b"

parameter passed to callback function when signal "c" changes
PLI_call_instrumented_line_num 0
assign_stmt_instrumented_line_num 20
lhs_signal "y"
rhs_expression "a || c"